Abstract Procedures and the Physical World

[Published in Proceedings of the AISB'13 Symposium on Computing and Philosophy, pp. 66-73, ISBN 978-1-908187-31-4]

Abstract Procedures and the Physical World

Paul Schweizer[1] and **Piotr Jablonski**[1]

Abstract. The paper examines some central issues concerning the notion of implementing abstract formal structures, including effective procedures and dynamical systems, in the realm of physical space-time. We address the view originally put forward by Putnam and Searle, that virtually any physical system can be interpreted as implementing virtually any computational formalism, and defend the general conclusion that realizing an abstract procedural structure is not an intrinsic property of physical systems, but rather is a purely observer-dependent ascription. In a parallel manner, the 'trivialization' arguments originally put forward against computationalism are extended to dynamical systems theory, an alternative abstract framework that has also been advocated as providing the theoretical foundation for mentality in the natural world. Rather than attempting to distinguish 'true' from 'false' cases of implementation, we distinguish *pragmatically useful* ascriptions from those that serve no epistemic purpose.

1 ENGINEERED IMPLEMENTATION

From a disembodied mathematical perspective, classical computation comprises an extremely well defined and stable phenomenon. Central to the theory of traditional computation is the intuitive notion of an effective or 'mechanical' procedure, and there are any number of different possible frameworks for filling in the details and making the idea rigorous and precise. Turing's 'automatic computing machines' [1] (TMs), supply a very intuitive and elegant rendition of the notion of an effective procedure, but there is a well known variety of alternative frameworks.

According to the widely accepted Church-Turing thesis, the class of computable functions is nonetheless captured in a mathematically absolute sense by the notion of TM computability, and every alternative formalization so far given of the broad intuitive notion of an effective procedure has been demonstrated to be equivalently powerful, and hence to specify exactly the same class of functions [2]. Thus the idealized notion of in-principle computability, where all finite bounds on input size, storage capacity and length of running time are abstracted away, seems to constitute a fundamental category, a stable and fundamental 'mathematical kind'.

A related further question is whether any sort of comparable feature carries over to computation as implemented or realized in the physical universe. Turing machines and other types of computational formalisms are *mathematical abstractions* and don't exist in real time or space. In order to perform *actual* computations, an abstract Turing machine must be realized by a suitable arrangement of matter and energy, and as Turing observed long ago [3], there is no privileged or unique way to do this. Like other abstract structures, Turing machines are *multiply realizable* - what unites different types of physical implementation of the same abstract TM is nothing that they have in common as physical systems, but rather a structural isomorphism expressed in terms of a higher level of description. Hence it's possible to implement the very same computational formalism using modern electronic circuitry, a human being executing the instructions by hand with paper and pencil, a Victorian system of gears and levers, as well as more atypical arrangements of matter and energy including beer cans serving as tokens of the symbol '1' and rolls of toilet paper serving as the tape.

Adopting the conventions introduced by Schweizer [4], let us call this 'downward' multiple realizability, wherein, for any given abstract structure or formal procedure, this *same* abstract structure can be implemented via an arbitrarily large number of *distinct* physical systems. And let us denote this type of downward multiple realizability as '↓MR'. After the essential foundations of the mathematical theory of computation were laid, the vital issue then became one of engineering – how best to utilize state of the art technology to construct rapid and powerful physical implementations of our abstract mathematical blueprints, and hence perform actual high speed computations *automatically*. This is a clear and deliberate ↓MR endeavour, involving the intentional construction of artefacts, painstakingly designed to follow the algorithms that we have created. From this top-down perspective, there is an obvious and pragmatically indispensible sense in which the hardware that we have designed and built can be said to perform genuine computations in physical space-time.

2 NATURAL COMPUTATION?

In addition to these comparatively recent engineering achievements, but presumably still members of a single underlying category of phenomena, various authors and disciplines propound the notion of 'Natural Computation' (NC), and invoke a host of indigenous processes and occurrences as cases in point, including neural computation, DNA computing, biological evolution, molecular and membrane computing, slime mould growth, cellular automata, ant swarm optimization, etc. According to such views, computation in the physical world is not merely artificial – it is not restricted to the devices specifically designed and constructed by human beings. Instead, computation is a seemingly ubiquitous feature of the natural order, and the artefacts that we have produced constitute only a

[1] Institute for Language, Cognition and Computation, School of Informatics, Univ. of Edinburgh, EH8 9AD, UK. Email: paul@inf.ed.ac.uk.

very small subset of the overall class of computational systems that inhabit the physical universe.

The disciplinary and terminological practices surrounding NC plainly invite a more thorough and rigorous examination of the underlying assumptions involved. Salient questions in need of scrutiny include: To what extent, if any, is computation a genuine *natural* kind – is there an intrinsic unity or core of traits systematically held in common by the myriad of purported examples of computation in the physical world? In what sense, if any, can computation be said to take place spontaneously, as a truly native, 'bottom-up' phenomenon?

The issue has pronounced conceptual importance with respect to positions on the conjectured computational nature of *mentality and cognition*. According to the widely embraced computational theory of mind (CTM), which underpins cognitive science, Strong AI and various allied positions in the philosophy of mind, computation (of one sort or another) is held to provide the scientific key to explaining and, in principle, reproducing mentality artificially. The paradigm maintains that cognitive processes are essentially computational processes, and hence that intelligence in the physical world arises when a material system implements the appropriate kind of computational formalism. So it's an immediate corollary of CTM that the human brain counts as an exemplary instance of natural computation.

Hence it is crucial to CTM's theoretical stance that there be a rigorous and precise analysis of physically grounded computation in the case of organically engendered human brains. But the issue has wider and independent significance, in an attempt to gain conceptual clarity on whether and to what extent computation can be cogently viewed as a natural occurrence. And this in turn requires a general theoretical investigation and articulation of what it means for computations and other sorts of abstractly specified formalisms and structures to be implemented in the physical realm. It is this last, overarching theme that will comprise the primary focal point of the ensuing discussion.

3 THREE DIFFERENT SENSES

For the sake of analytical precision, we will begin by disambiguating three possible senses in which real physical systems might be thought of as 'performing a computation', and where these distinct senses are often blurred or run together by proponents of NC.

First (1), a physical system or object may be said to *obey or satisfy* a particular equation or mathematical function. For example, a falling body in the earth's gravitational field will satisfy or obey Newton's equation for gravitational acceleration. Similarly, the planets orbiting the sun satisfy or obey Kepler's laws of planetary motion. This has lead various NC enthusiasts to claim that the planets orbiting the sun, falling bodies in the earth's gravitational field, etc., are in fact *computing the values* of the equations in question. Taken to its most extreme form, this becomes the assertion that physical processes and natural laws are themselves fundamentally *computational*, and hence that computation constitutes the foundational key to the natural order.

Second (2), the activities of a physical system or process may be precisely *modelled or simulated* by a given computational formalism or depiction. For example, it is possible to create highly accurate and explanatorily useful computer models which simulate the behaviour of various complex physical events such as earthquakes, climate change, hurricanes, particle collisions, protein folding, brain processes, etc. Again, the usefulness and accuracy of these computational models has lead proponents of NC to claim that the physical phenomena *themselves* are performing such computations or are somehow instances of such computations occurring in nature.

And third (3), a physical device or process may be said to literally *implement, realize or execute* a particular algorithm or effective procedure. Thus when I write a piece of code in some artificial programming language, say Prolog, and then run this code on my desktop computer, there is a very clear and paradigmatic sense in which the electro-mechanical hardware in question is performing or executing the algorithm explicitly encoded in Prolog.

It seems uncontroversial that (3) is the basic and indeed canonical sense of computation in the physical world, and constitutes the modern historical origin of the concept. But in the Prolog example used to illustrate the import of (3), the computation in question is not a natural occurrence ─ rather it's a direct result of human design and engineering. Human artefacts in the form of electromechanical hardware devices comprise the arrangements of matter and energy that carry out the actual computation in space and time, and the procedures being executed are specified in terms of artificial programming and machine languages. In such literal and exemplary cases, real world computation is a purely synthetic phenomenon.

However, this does not in itself rule out the possibility that there could be genuine *natural* computation in the stringent sense of (3), since it is still entirely possible that some appropriate version of CTM is true. For example, if Fodor [5] is correct, then the human brain, an organically engendered 'wetware' device, is running the Language of Thought (LOT) as an indigenous formal system of rule governed symbol manipulation, in a manner directly comparable with a computational artefact. Thus if Fodor is correct, then the human brain is a paradigmatic instance of NC in sense (3).

Accordingly, in the ensuing discussion we will treat Fodor's conjectured LOT as epitomizing what genuine computational processing in the natural world would look like in its most explicit form − a spontaneously generated, bottom-up case of formal symbol manipulation. And this is compatible with the foregoing discussion of downward multiple realization, since the relation between LOT and the brain could then be viewed in typical ↓MR terms. For example, an alternative mechanical device, physically quite unlike the brain, could presumably be constructed to implement the LOT in an artificial medium.

In order to construct such an implementation, we would need to utilize expertise in engineering and materials science, which exploited the lawlike regularities that characterize the time evolution of physical systems. This expertise would be required to design the material implementation in such a way that it could be methodically interpreted, at the appropriate level of description, as behaving in a manner isomorphic to the abstract processing structure of the human brain. Indeed, it's perfectly conceivable that if we could abstract out the relevant computational structure of the LOT physically realized in brain activity, then we could run this same abstract computational structure on some version of our existing artificial hardware. And then, in perfect accord with ↓MR, the systematic and predictable behaviour of both the brain and the artificial device, seen as systems governed by natural law, could be interpreted, *at*

a higher level of description, as implementations of the same abstract computational structure.

4 CRITIQUE OF SENSES (1) AND (2)

As noted in sense (1) above, a physical system or object, such as a piece of electromechanical hardware, may be described as *obeying or satisfying* various equations or mathematical functions. And it is by utilizing our knowledge of these regularities that we are able to construct physical realizations of abstract computational procedures, and thereby systematically and reliably preserve the implementational mapping from abstract formalism to relevant sequences of states of the physical machine. What then is to be gained by then claiming that, in addition to performing a computation in virtue of systematically preserving this mapping, the hardware is performing yet *another*, underlying computation, simply in virtue of evolving through time in accordance with natural laws?

Such a claim seems to be founded on a conflation between two of Marr's [6] classic distinctions in levels of analysis. The salient difference between what is going on in (1) as opposed to (3) is precisely the difference between the level of bare mathematical function and the level of computational algorithm. For any given function there are many different algorithms for computing its input/output values, and a mathematical function or general equation on its own does not specify any corresponding formal method or single out any one of the many different possible corresponding effective procedures as privileged. Hence merely *satisfying an equation*, as in the case of a hardware device obeying a lawlike physical regularity, is too weak to underwrite an assertion of distinctively *computational* processing, because it leaves the vital procedural details completely unspecified. Exactly *which* particular algorithm for computing the values of Kepler's laws of planetary motion is the earth currently implementing? And articulated in precisely *which* abstract computational framework?

Thus sense (1) seems to constitute an unmotivated and theoretically unilluminating inversion of perspective. Human scientists have devised mathematical abstractions in the form of general equations in order to characterize observed regularities in physical behaviour. In turn, we utilize these rigorously characterized regularities to construct artefacts that can be systematically interpreted, at a higher level of description, as implementations of selected computational formalisms. In this respect, we have a well defined implementational foundation in brute physical behaviour. The ontological and causal status of real-time computations is thus grounded in a stable, non-computational medium. But to then assert that the implementational medium *itself*, simply in virtue of evolving in accord with certain abstractly characterized patterns, is thereby performing lower level computations, seems to threaten a causal/ontological regress. And unless the particular algorithms and formalisms purportedly being executed are explicitly specified and substantiated, the assertion seems to make no additional contribution. However, the general equations as such are central to our scientific theorizing and abstract representation of the natural world. In section 7 below we shall further investigate sense (1) and the status of such formal specifications in the particular guise of dynamical systems.

In the case of sense (2), the algorithmic details missing from sense (1) are provided, but they are located in the wrong place. When complex physical events and processes are accurately simulated via computational models, it is the *artificial* computational structures which compute the values of the laws, equations and regularities governing the physical phenomena being simulated. And indeed, this is why the *models* are accurate and useful. But what motivates the further claim that the complex physical phenomena are *themselves* somehow implementations of the computations performed by the artificial models? Again, the same equations and regularities could be computed by another computational model using different underlying algorithms, programming languages, etc. to calculate the relevant values. Which of the many distinct computational possibilities is privileged or singled out by nature? In agreement with Piccinini [7], we would advocate a sharp distinction between mere computational *modelling* and genuine computational *processing* in nature.

So in the ensuing discussion we will treat (3) as the literal and canonical sense of computation in the space-time arena. We diagnose sense (1) as derived from the mathematical characterization of fundamental regularities in nature, but where the additional attribution of computational activity is due to a conflation between Marr's distinct levels of bare mathematical function versus specific algorithm for computing the values of the function. Finally, sense (2) is a case of artificial *simulation* of natural events and processes, where the values of the regularities salient to sense (1) are explicitly computed, but where this computation is merely a tool of human heuristics and is not supported by nature. In this respect sense (2) is a hybrid of the more basic content involved in (1) and (3), and will not receive any further investigation. The ensuing discussion will focus primarily on (3) as the paradigmatic sense of computation in the physical world, and will also provide an allied investigation of sense (1), since both (1) and (3) are cases of applying explicit renditions of abstract, formal procedures directly to physical events and processes.

Of course, to some extent the issue could be seen as purely terminological. One *could* choose to brand computation in sense (3) as 'classical' or 'Turing' computation, and then label senses (1) and (2) as 'computation', but of a different, broader sort. But for this broadening of the scope of application of the term to count as useful and well motivated, it would need an accompanying story to explain (i) what essential characteristics are had in common to unify all three apparently quite disparate senses of the term, and then (ii) why the category of phenomena so unified should be called 'computation' and not something else.

We don't wish to dwell on mere terminological or taxonomical disputes, and hence maintain that, whatever use of terminology one may adopt, sense (3) has clear and paradigmatic import, and it is this interpretation of the word that we wish to emphasize and investigate. Furthermore, whatever may be going on in most cases of (1) and (2), be it felicitously categorized as 'natural computation' or as something else, it is still quite distinct from what is captured by sense (3).

5 COMPUTATION IS NON-INTRINSIC

We will now articulate and begin to defend one of the main theses of the paper, a thesis stemming from arguments originally put forward by Putnam [8] and Searle [9, 10], that even in the quite restricted and canonical sense of (3), computation is not an inherent or intrinsic characteristic of any physical system.

Instead, it's a purely *observer dependent ascription*, projected onto a physical system via an act of human interpretation. Furthermore, the extent to which a physical device can be interpreted as realizing any sufficiently rich computational formalism, such as an abstract Turing machine, is not absolute, but instead is always a matter of degree of approximation. And the choice to interpret a physical device as implementing a particular abstract formalism is always relative to our particular purposes and potential epistemic gains.

Our normal practice of interpreting specialized artefacts as performing computations is clearly of very high pragmatic value. Nonetheless, such interpretations are ultimately dependent on human conventions and are not intrinsic to the hardware itself. Thus computation in the physical world is not sustained or underwritten by the innate structure of the systems interpreted as realizers, and computation as such is not a natural kind. We will begin our defence of this view by examining some well known arguments concerning the theoretical possibility of multifarious 'deviant' interpretations.

6 TRIVIALIZATION ARGUMENTS

Various critics of CTM have put forward a family of 'trivialization arguments', directly relevant to sense (3) above. The arguments are based on the contention that the notion of a physical system implementing a computational formalism is overly liberal to the point of vacuity. As a case in point, Putnam [8] offers a proof of the thesis that *every* open physical system can be interpreted as the realization of *every* finite state automaton. Putnam's argument will be explored in more detail in section 7, in the context of Dynamical Systems theory.

In the current section of the paper we will consider the closely related position advanced by Searle [9], who argues that virtually any physical system can be interpreted as following virtually any program. Thus hurricanes, our digestive system, the motion of the planets, even an apparently inert lecture stand, all possess a level of description at which they instantiate any number of different abstract formal procedures. The stomach has inputs, internal processing states and outputs, and if one wanted to, one could interpret the inputs and outputs as code for any number of different symbolic processes. And in [10] Searle attempts to illustrate the extreme conceptual looseness of the notion of implementing an abstract formalism by famously claiming that the molecules in his wall could be interpreted as running the WordStar program.

Again adopting conventions introduced by Schweizer [4], let us label multiple realizability in this direction, wherein any given *physical system* can be interpreted as implementing an arbitrarily large number of different *computational formalisms* 'upward MR' and denote it as '↑MR'. The basic import of ↑MR is the *non-uniqueness* of computational ascriptions to particular physical systems. In the extreme versions suggested by Putnam, Searle, and more recently Bishop [11], there are apparently no significant constraints whatever – it is possible in principle to interpret every open physical system as realizing every computational procedure. Let us call this extreme version 'universal upward MR' and denote it as '↑MR*'. Mere ↑MR is weaker than ↑MR*, since the former does not assert that there are no salient constraints, and hence ↑MR would be consistent with the denial that, e.g., the molecules in Searle's wall can in fact be interpreted as implementing the WordStar program, although every physical system is still interpretable as implementing some very large set of distinct computations.

In the present discussion we will not argue for or against ↑MR* but instead confine our considerations to the more modest ↑MR. In view of ↑MR, it's still never the case that any given computational interpretation of a physical system is privileged or unique, and this is far more difficult to deny than the powerful and broad sweeping ↑MR*. In turn, the non-intrinsic status of computation would seem to follow as a direct consequence of mere ↑MR alone. As long as there are at least two distinct interpretations, there is no objective fact of the matter regarding *which* computation is 'actually' being performed, nor which of the alternatives is the 'correct' or 'real' account. And this is because the computation itself is not an intrinsic property of the physical device, and is instead dependent on a human observer to supply the various alternative interpretations.

This is not to say that it's purely a matter of caprice, and that there are no objective constraints that the interpretation must satisfy. Instead, the situation is perhaps comparable to the distinction between natural kinds, such as water, and conventional kinds, such as being a table. Even though membership in either kind might be based on criteria whose satisfaction (or not) is a matter of objective truth, still the criteria for conventional kinds are not intrinsic, and there is nothing about the particular arrangement of matter now holding up my desk top computer which makes it intrinsically a table. The salient criteria stem purely from human practices and stipulations rather than from, e.g., fundamental microstructure or natural law.

The original trivialization arguments are intended to undermine CTM, by showing that attributions of computational processing are overly liberal to the point of vacuity, and hence cannot serve as a criterion for mentality in the natural world. But the potential scope of application is clearly much wider, and they also serve to trivialize the idea of 'Natural Computation' in general. According to ↑MR*, anything computes everything, and hence computational processing in the natural world turns out to be far more rampant and ubiquitous than proponents of NC ever suspected.

7 SYNTAX, SEMANTICS, PHYSICS

At the abstract, formal level, computation is an essentially syntactic phenomenon, and how we choose to interpret arrangements of matter and energy as constituting, say, tokens of an abstract syntactic type, and thus specifying an implementation of the basic computational vocabulary, is entirely independent of physical composition. For example, in the downward ↓MR direction there is a more or less limitless diversity in the ways in which material patterns and arrangements can be viewed as implementing the binary notation of '0' and '1', from ink marks on a piece of paper, stones placed in wooden boxes, patterns on old-fashioned punch cards, electric voltages, beer cans positioned on rolls of toilet paper, … And this applies in the reverse ↑MR direction as well, wherein the same stones placed in wooden boxes can be interpreted as implementing any number of distinct computational formalisms.

Classical computation is rule-governed syntax manipulation, and it is no more intrinsic to physical configurations than is syntax itself. It is also worth observing

that discrete states are themselves idealizations, since the physical processes that we interpret as performing computations are in fact continuous, and we must abstract away from the continuity of the underlying substrate and impose a scheme of conventional demarcations to attain discrete values. Hence even this elemental building block of digital procedures must be projected on to the natural order from the beginning. The irresistible conclusion to be drawn is that there is a fundamental gap separating 'concrete' physical reality from the human-based ascriptions of abstract syntactic features.

In turn, there is yet *another* fundamental gap separating abstract syntactic features from their semantic interpretation. Just as syntax is not intrinsic to physics, so too semantics is not intrinsic to syntax. Just as being an instance of the spoken English sentence 'The cat is on the mat' is not an inherent property of the sound waves constituting any particular utterance token, so too, the associated proposition comprising the *interpretation* of the utterance is not intrinsic to the abstract syntactic structure. Instead, the associated meaning is determined via arbitrary human convention, and the same syntactic item could just as well have had the interpretation currently expressed in English by 'The rat is on the table' or 'The dog is on the hearth'.

In the context of classical computation, one of the key constraints in the notion of an effective procedure is that the rules can be followed 'mindlessly', i.e. without knowing what the manipulated symbols are supposed to *mean*. As a consequence, there is no unique meaning determined by the procedure as such, and a multitude of distinct and *incompatible* interpretations are always possible. As a simple example, a Turing machine intended to compute the values of a particular truth function, say inclusive disjunction, can be easily reinterpreted as computing conjunction instead, simply by flipping our interpretation of the symbols '0' and '1', so that '0' is construed as denoting true while '1' denotes false. And the same procedure interpreted as computing conjunction could instead be construed as computing the values of the arithmetical function of multiplication. restricted to the numerical inputs 0 and 1.

Similarly, formal systems in general are such that the transformations on symbols are not specified with reference to their intended interpretation. Many classical *negative* results in mathematical logic stem from this separability between formal syntax and meaning. The various upward and downward Löwenheim-Skolem theorems show that formal systems cannot capture intended meaning with respect to infinite cardinalities. As another eminent example, Gödel's incompleteness results involve taking a formal system designed to be 'about' the natural numbers, and systematically reinterpreting it in terms of its own syntax and proof structure. As a consequence of this 'unintended' interpretation, Gödel is able to prove that arithmetical truth, an exemplary *semantical* notion, cannot, in principle, be captured by finitary proof-theoretic means.

In summary, there are *two* fundamental gaps separating formal procedures, standardly interpreted as computing the values of given functions, from the physical processes that we construe as implementing such procedures. First there is the gulf dividing the intended semantic interpretation from the bare syntactic formalism, and second there is the chasm between abstract syntactic formalism and physical reality. In *both* cases the gaps can only be bridged by an act of purely conventional human interpretation. And it is in this sense that computation in the physical world is inherently observer dependent.

8 PUTNAM AND DYNAMICAL SYSTEMS

We will now take a closer look at the foregoing sense (1) sometimes used to support the view of computation in nature. As was noted in section 3, the term 'computation' is occasionally meant as a description of the fact that some physical processes *satisfy* or *obey* a mathematical equation. Although, as we noted, there is little reason to believe that planets orbiting the sun compute an algorithmic approximation of the Newtonian laws of motion. So instead of viewing such cases in explicit NC terms, we will analyze the underlying notion of projecting the bare, abstract procedural descriptions furnished by dynamical systems onto the time evolution of physical phenomena. Thus, instead of a mapping between the physical states of the system and the computational states of an algorithm, there is a mapping between physical states and the variables of the appropriate dynamical system (DS). Following Jablonski [12], we argue that DS, when viewed in this manner, exhibit striking similarities to the inputless Finite-State-Automata (iFSA) analyzed by Putnam [8], and that the endeavour succumbs to very similar difficulties as originally proposed by Putnam in the context of strong ↑MR* arguments against computationalism.

Since an iFSA is defined by the set of monadic computational states and the rules of transitions from any given state to the next, any execution of an algorithm will take the form of the sequence of states. Putnam has noted that the evolution of any non-cyclic physical phenomena can be divided into a sequence of periods in such a way that we can map the physical states of the system onto the computational states of an iFSA. Say, we want to show that Searle's wall realises a computation performed by the iFSA defined by the states A, B and C and rules of transition A>B, B>C and C>B. If initialised in the state A the automaton will transit through states ABCBCB and will continue to oscillate between states C and B. We can claim that the wall performed 6 steps of the computation within any period of time e.g. from 12:00 to 12:06. We simply label all physical states of the wall within the first minute as computational state A, within the second minute as state B, third – C, forth – B, fifth – C and sixth – B.

Since the complexity of the thermal movements in the wall, openness of the system and, possibly, some non-reversible physical phenomena (e.g. radioactive decay) insure that every physical state of the wall will manifest itself only once, the labelling will be functional i.e. for every physical state only one computational state is given (however, a single computational state can be realised by many different physical states).

In applying this same basic strategy using a Dynamical Systems framework rather than inputless Finite-State-Automata, we first note that a system is defined by the set of its variables and rules governing the evolution of the variables over time. Unlike iFSA that have a finite number of possible states, the states of a DS are given by the vector of real number variables. Thus DS have an infinite number of possible states. Usually, the dynamical models are defined as deterministic, continuous systems so the rules governing the dynamics are given by a set of differential equations. Equations are the continuous equivalents of the rules of transitions for the iFSA. Finally, the phase-space trajectory of the system, the evolution of its

variables over time, can be compared to the sequence of computational states of the iFSA.

The analogies between the two formalisms can be summarised as follows:

iFSA	DS
Finite number of states	Infinite number of states
Initial state	Initial conditions
Table of transitions	Differential equations
Computational steps	Time
Sequence of states	Trajectory through the phase space

Using these analogies, we can see that it is possible to map any finite trajectory of any DS onto any non-cyclic physical process in a manner similar to Putnam's original strategy. We map the first point of the trajectory onto the first physical state of the system and all consecutive points onto the corresponding physical states.

First, we need to map a real, physical time of the process onto the abstract time of the DS. Since we are interested in the finite period, we may define a mapping function *M* as follows:

$$\tau = M(t) = \left[\frac{t - t_0}{t_e - t_0}\right] * [\tau_e - \tau_0] + \tau_0$$

where τ is an abstract time of the DS, t – real, physical time, τ_0 – the beginning of the dynamical process, τ_e – the end of the dynamical process, t_0 and t_e – the beginning and the end of the physical process in real time.

We need to include Putnam's condition of non-cyclic behaviour of the physical system in order to guarantee, that every moment of the physical time t indicates just one physical state s_t of the system. Given that DS is defined by its equations of motion

$$\frac{dx}{d\tau} = g(x)$$

where x is a multidimensional vector of variables, and a single trajectory of the DS is determined by the initial conditions x_0 and the time interval, we can take an integral of the equations of motion that will have the form of the function of the state of the DS over time $x = h(\tau)$. This integral defines the trajectory of the system in phase space and is the equivalent to the sequence of states in the case of iFSA. However, since the trajectory is composed of the infinite number of points it has to be expressed as a function of time and cannot be presented in a form of a table of values. Now we can form a labelling function *f* that assigns abstract states of the DS to the physical states s_t of the system.

$$f(s_t) = h(M(t))$$

Thus for every physical state s_t we know the time *t* when it appeared (since the behaviour of the system is non-cyclic). Knowing the time *t* and the time-mapping function *M(t)* we can determine the corresponding time $\tau = M(t)$ of the abstract dynamics. Eventually, since we know the states of the DS as a function *h* of time τ we can determine the formal state x of the DS, and what follows, the values of its variables. In other words, for every non-cyclic physical system and finite period of time we are able to map its states onto states of any DS.

A main difference between DS and iFSA lies in the fact that the latter have a finite number of states and steps while the former have an infinite number. However, since physical systems are continuous in nature the mapping still can be carried out. In order to perform Putnam's version of trivialization we need to know *in advance* the sequence of the states of the iFSA. In the case of DS we ought to know the trajectory of the system. That was easy for iFSAs since computation of the sequence required only a finite number of steps. However for the DS we need to integrate the equations of the DS. In many cases, nonlinear differential equations do not have analytic solutions so we are unable to obtain the trajectory of the system in the form of a function of variables over time.

This limitation is however, only epistemic in nature. Every DS has a trajectory defined for every initial condition even if we are unable to discover its analytic form. We still may conclude that, in principle, there is a mapping between any DS and physical system, although in many cases we will unable to provide the specific details.

9 CONSTRAINTS ON IMPLEMENATION

In response to ↑MR* and the trivialization arguments, various authors, including Chrisley [13], Chalmers [14], Copeland [15], and Block [16] have proposed a number of constraints on computational interpretations in an attempt to distinguish 'true' cases of implementation from the myriad of purportedly 'false' cases utilized by Putnam and others. Two of the most intuitively compelling restrictions are supplied by (i) causal and (ii) counterfactual considerations. The first constraint holds that the pattern of abstract state transitions constituting a particular run of the computational procedure on a particular input must map to an appropriate transition of physical states of the machine, where the relation between succeeding states in this physical sequence is governed by proper causal regularities. The second constraint holds that a necessary condition for being a 'genuine' implementation is the ability of the mapping to support counterfactual sequences of transitions on inputs not actually given. This constraint is prompted by the fact that various ↑MR* mappings from formalism to physical system, given by Putnam and others, are defined only for a single run and say nothing about what would have happened if a different input had been given (see Bishop [11] for an exception).

Although both (i) and (ii) are intuitively plausible suggestions, we view both as ultimately unsuccessful in blocking the trivialization results. See Schweizer [4] for arguments to the effect that, within the context of computation in canonical sense (3), constraint (i) provides a sufficient but not a necessary condition, while (ii) is unsatisfiable in principle (for sufficiently rich frameworks, such as those invoked in the Church-Turing thesis), and can serve only as a measure of degree of approximation.

In the context of 'computation' in sense (1), and pertinent to the above extension of the ↑MR* arguments to Dynamical Systems, we will now briefly examine two of the main points raised by Chalmers [14] and address them in the context of DS trivialization. In line with (ii) above, it has been objected that Putnam's labelling does not map counterfactual computational states onto any physical states. If a given iFSA is defined by states A, B, C, D and transitions A>B, B>C, C>B and D>B it could also perform the sequences ABCBCB, BCBCBC, CBCBCB or DBCBCB. Because Putnam's method of construction of the labelling function only works for the single chain of physical events, we would be unable to map state D

onto any physical state of the wall because D did not appear in the sequence used for labelling.

Since an iFSA can perform only a finite number of state sequences, we need to form a labelling function that maps all those possible sequences onto some states of the physical system and show that such a labelling can be constructed for an arbitrary physical device or phenomenon. In the context of classical computation, this stronger version of trivialization is known as the "*clock and dial*" reply to Chalmers objection. The argument states that every physical system can contain not only non-reversible physical phenomenon (a *clock*) but also some physical magnitude that can be set into a number of distinct states and will remain in the same state for the given period of time (a *dial*). Thus the complete state of the system is defined by the pair [$\boldsymbol{d}$, S_t], where $\boldsymbol{d}$ is the state of the dial and S_t – the state of the clock at time *t*. Since our iFSA can perform four different sequences, $\boldsymbol{d}$ will take four values. The state of the dial will determine which sequence is mapped onto the states of the *clock*. Thus if the dial is set to d_1 we will map the first sequence onto physical states of the system during this time period. If the dial is set to d_2 we will map the second sequence and so on. One can argue that not every physical system contains *clock* and *dial* components, however there is certainly a large class of such systems thus the '↑MR' is conserved.

A parallel strategy can be applied in the context of DS trivialization. The infinite number of possible trajectories of the DS forces us to modify the "*clock and dial*" response to the argument. The *clock* will have to transit continuously through an infinite number of physical states within a finite amount of time (which seems to be uncontroversial since physical time is continuous) while the *dial* will have to be substituted by devices that can be set in an infinite number of states. We may picture the devices as set of continuous *sliders*, one for every variable in the equations of the DS. Every initial state of the DS can be encoded by the appropriate setting of the *sliders* just as every initial state of the iFSA can be encoded as a position of the *dial*. After that, we map the integrated trajectory of the DS onto the run of the *clock*.

A second requirement is reliability, which is closely akin to causal regularities − a proper labelling function should interpret not only one but every evolution of the physical system from the same initial state as a realisation of the same algorithm. If Searle is right, then he should be able to reliably and repeatedly reset his wall to a given initial condition and demonstrate that its physical evolution is identical to the one used for the initial labelling. Since the wall is an open system and (as required for the trivialisation argument) exhibits non-cyclic evolution it certainly will not repeat its states.

However, the "*clock and dial*" version of the argument seems to be immune to this objection. The dial can be reliably set into any of its states and the *clock* will reliably pass through the same sequence of moments. And it appears that there will be no significant difference in this regard between a continuous "*clock an dial*" and the sequential counterpart used for iFSA as long as we are able to demonstrate that the *clock* can reliably pass through its sequence of states and the *sliders* can stay in the unchanged position through the period of observation.

10 COMPUTATION AND PRAGMATICS

We would now like to propose a different perspective on the issue. Rather than distinguishing 'true' from 'false' cases of implementation, what these and other proposed constraints do instead is to go some distance in distinguishing interesting, conceptually rich and *pragmatically useful* implementations from the many uninteresting, trivial and useless cases that abound in the space of possibility. It's certainly true that there is no pragmatic value in most interpretive exercises compatible with ↑MR and ↑MR*. Ascribing computational activity to physical systems is *useful* to us only insofar as it supplies *informative outputs*, which in most cases will come down to new information acquired as a result of the implemented calculation.

So, interesting and useful observer dependent computation takes place when we can directly read-off something that *follows from* the implemented formalism, but which we didn't already know in advance and explicitly incorporate into the mapping from the start. That's the incredible value of our computational artefacts, and it's the only *practical* motivation for playing the interpretation game in the first place. Hence a crucial difference between our computational artefacts and the attributions of formal structure to naturally occurring open systems, as employed by ↑MR* exercises, is that the mapping in the latter case is entirely *ex post facto* and thus supplies us with no epistemic gains. The abstract procedural 'trajectory' is already known and used as the basis for interpreting various state transitions in the open system and hence characterizing it as an implementation. In sharp contrast, we can use the intended interpretation of our artefacts both to *predict* their future behavior, as well as *discover* previously unknown output values automatically.

And this is obviously why an engineered correlation obtains between fine-grained causal structure and abstract formal structure in the case of our artefacts – we want them to be informative and reliable! We also want them to be highly versatile, and this is where counterfactual considerations come to the fore in practice: over time we can do runs on a huge number of different inputs, and in principle the future outputs follow as direct consequences of the intended interpretation. So a physical system is *useful to us* as a computer only when its salient states are distinguishable by us with our measuring devices, and when we can put the system into a selected initial state to compute the output of our chosen algorithm on a very wide range of specific input values.

These *pragmatic* considerations supply clear and well motivated criteria for differentiating useful from useless cases of physical implementation. And we would advocate this type of pragmatic taxonomy in lieu of attempts to give overarching theoretical constraints purporting to distinguish 'true' from 'false' cases. Some basic desiderata for pragmatically valuable implementations include (a) fully automatic, (b) reliable, (c) versatile in the sense of computing values for a wide range of different inputs (d) non *ex post facto* (e) yielding increased predictive power with regard to future physical states of the implementing mechanisms, (f) possessing technologically manipulable initial configurations and output configurations detectable by our measuring devices and (g) physical rather than purely abstract constraints on the input and output characterizations.

Similarly, ascriptions of computation in the sense (1) to the physical systems are motivated by pragmatic reasons. Useful interpretations ought to yield simple formalisms whose

equations we are able to integrate or at least investigate their properties with our mathematical resources. Variables should be mapped onto reliably observable physical magnitudes that figure in many scientific theories and are not proposed ad hoc. Valuable interpretations of physical phenomena in terms of DS give us simplified formal description, epistemically useful and, hopefully, manifesting some predictive powers. We cannot however, claim that the physical process itself realises the mathematical equations in any other sense than that its behaviour can be fruitfully described using such equations.

11 CONCLUSION

Computation is an extrinsic, observer dependent interpretation that we project onto physical systems according to our purposes and potential epistemic gains. As such, it does not support a stable or independent natural kind. Diverse types of natural events and processes can be modelled or simulated using computational techniques, as in sense (2) above, but this is to be distinguished from canonical sense (3), in which the system *itself* is viewed as instantiating and executing an explicit formal procedure. However, various physical systems *do* spontaneously 'obey' clear regularities in their evolution through time, and many such regularities have been mathematically characterized in terms of Dynamical Systems theory. Although this sense (1) reading does not comprise a case of genuine *computation*, in the strict connotation of executing a well defined formal procedure, it does provide a fundamental form of mathematical representation of the natural world.

Various opponents of the Computational Theory of Mind have provided trivialization arguments to the effect that, even in canonical sense (3), the notion of implementing a computational formalism is overly liberal to the point of vacuity. Such results serve to undermine not only CTM in particular, but the more encompassing notion of 'Natural Computation' in general. In a parallel manner, we extend this strategy to sense (1) 'computation' in the guise of Dynamical Systems theory, to argue that realizing such abstract formal structures is again a matter of observer dependent ascription. As with the original trivialization strategies aimed against CTM, this extended result has deep implications for the science of mind, since Dynamical Systems have been advocated (by, e.g. van Gelder [17]) as providing an alternative theoretical foundation for mentality in the natural world.

Advocates of CTM have proposed a number of constraints on the notion of 'genuine' implementation, in an attempt to block the trivialization results and uphold a robust notion computation in the physical world. However we argue that such constraints derive purely from human interest as opposed to underlying and independent matters of fact. Rather than serving to distinguish true' from 'false' cases of implementation, what the proposed constraints do instead is to help distinguish conceptually rich and *pragmatically useful* implementations from the many uninteresting, trivial and useless cases that abound in the space of theoretical possibility.

Although practical considerations clearly guide the design and construction of our computational artefacts, such pragmatic motivations do not justify any deep or ontologically grounded distinction between genuine versus trivial interpretations. Hence we support an anti-realist view of computation in nature, and implementing or realizing abstract formal structures in general is not an intrinsic property of physical systems. In particular we have viewed the notion of implementation in the context of senses (1) and (3), but the view generalizes to all the prospective forms of non-Turing 'computation' inspired by considering natural events and processes. These are abstract, observer dependent ascriptions projected onto a more basic physical substrate.

REFERENCES

[1] Turing, A., 'On Computable Numbers, with an Application to the Entscheidungsproblem', *Proceeding of the London Mathematical Society*, (series 2), 42, 230-265, (1936).

[2] Boolos, G., Burgess, J.P. and Jeffrey, R.C., *Computability and Logic*, 5th edition, Cambridge University Press, (2007).

[3] Turing, A., 'Computing Machinery and Intelligence', *Mind*, 59: 433-460 (1950).

[4] Schweizer, P., 'Physical Instantiation and the Propositional Attitudes', *Cognitive Computation*, 4: 226-235 (2012).

[5] Fodor, J., *The Language of Thought*, Harvard University Press, (1975).

[6] Marr, D., *Vision*, CA: W. H. Freeman, (1982).

[7] Piccinini, C., 'Computational Modelling vs. Computational Explanation', *The Australasian Journal of Philosophy*, 85(1), 93-115, (2007).

[8] Putnam, H., *Representation and Reality*, MIT Press, (1988).

[9] Searle, J., 'Minds, Brains and Programs', *Behavioral and Brain Sciences* 3: 417-424, (1980).

[10] Searle, J., 'Is the Brain a Digital Computer?', *Proceedings of the American Philosophical Association*, 64, 21-37, (1990).

[11] Bishop, J. M., 'Why Computers Can't Feel Pain', *Minds and Machines*, 19, 507-516, (2009).

[12] Jablonski, P., *Trivialisation Arguments Against Dynamical Hypotheses*. MSc dissertation, University of Edinburgh (2012).

[13] Chrisley, R. L., 'Why Everything Doesn't Realize Every Computation', *Minds and Machines*, 4, 403-420, (1994).

[14] Chalmers, D. J., 'Does a Rock Implement Every Finite-State Automaton?', *Synthese*, 108, 309-333, (1996).

[15] Copeland, J., 'What is Computation?', *Synthese*, 108:335-359, (1996)

[16] Block, N., 'Searle's Arguments against Cognitive Science'. In J. Preston and J. M. Bishop *Views into the Chinese Room*, Oxford University Press, (2002).

[17] van Gelder, T. J., 'What Might Cognition Be If Not Computation?', *Journal of Philosophy*, 91: 345-381, (1995)